

Распределенные вычислительные системы

Лекция №9: Распределенные объектные транзакции

Алексей В. Бурдаков, к.т.н.
burdakov@usa.net

План лекций

№	Дата	Тема
1	04.09	Вводная лекция
2	11.09	Эволюция распределенных технологий
3	18.09	Принципы ПО среднего слоя
4	25.09	Стандарты OMG, CORBA и ORB
5	02.10	Пример приложения на CORBA
6	09.10	COM, Java/RMI
7	16.10	Определение местонахождения
	23.10	Перенос
8	30.10	Долговременное хранение + <u>Рейтинг по Л.1-7</u>
9 /		
10	06.11	Долговременное хранение

План лекций (продолжение)

№	Дата	Тема
11	13.11	Распределенные объектные транзакции
12	20.11	Лекция
13	27.11	Лекция
14	04.12	Лекция
15	11.12	Лекция
16	18.12	Лекция
17	25.12	Зачет

План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Долговременное хранение объектов

Проблема

- Что произойдет если отказ случится в процессе модификации ресурса?
 - Какие операции были завершены?
 - Какие не завершены и д.б. повторены?
 - В каком состоянии будут находиться ресурсы?
-
- «Идея распределенных систем без управления транзакциями подобна обществу без законодательства. Кто-то не ощущает необходимости в законах, но все нуждаются в способе разрешения конфликтов.»

Jim Gray

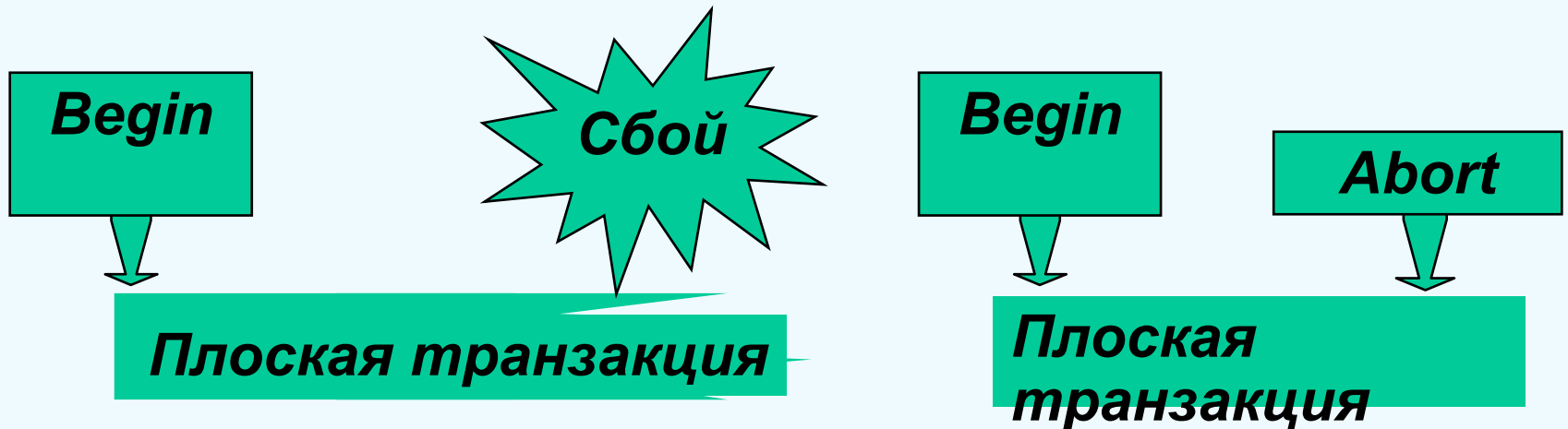
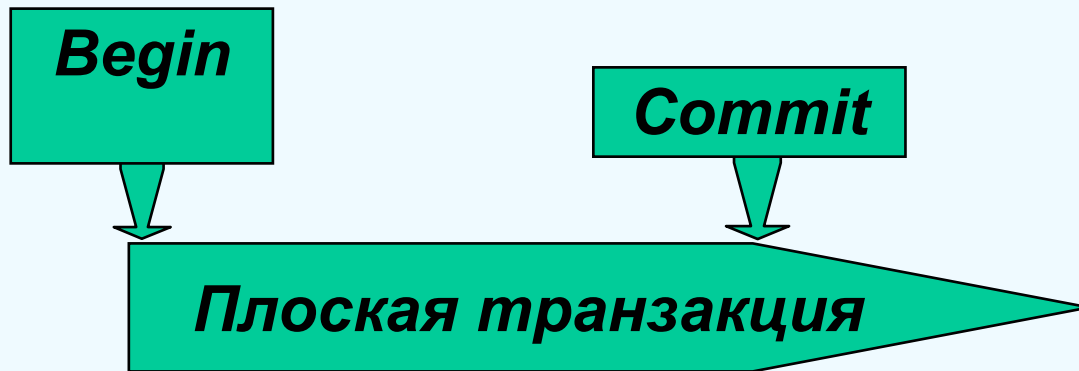
Концепция транзакций

- ACID
 - Atomicity
 - Consistency
 - Isolation
 - Durability
- Фиксация и откат транзакций
- Плоские и вложенные транзакции
- Централизованные и распределенные транзакции

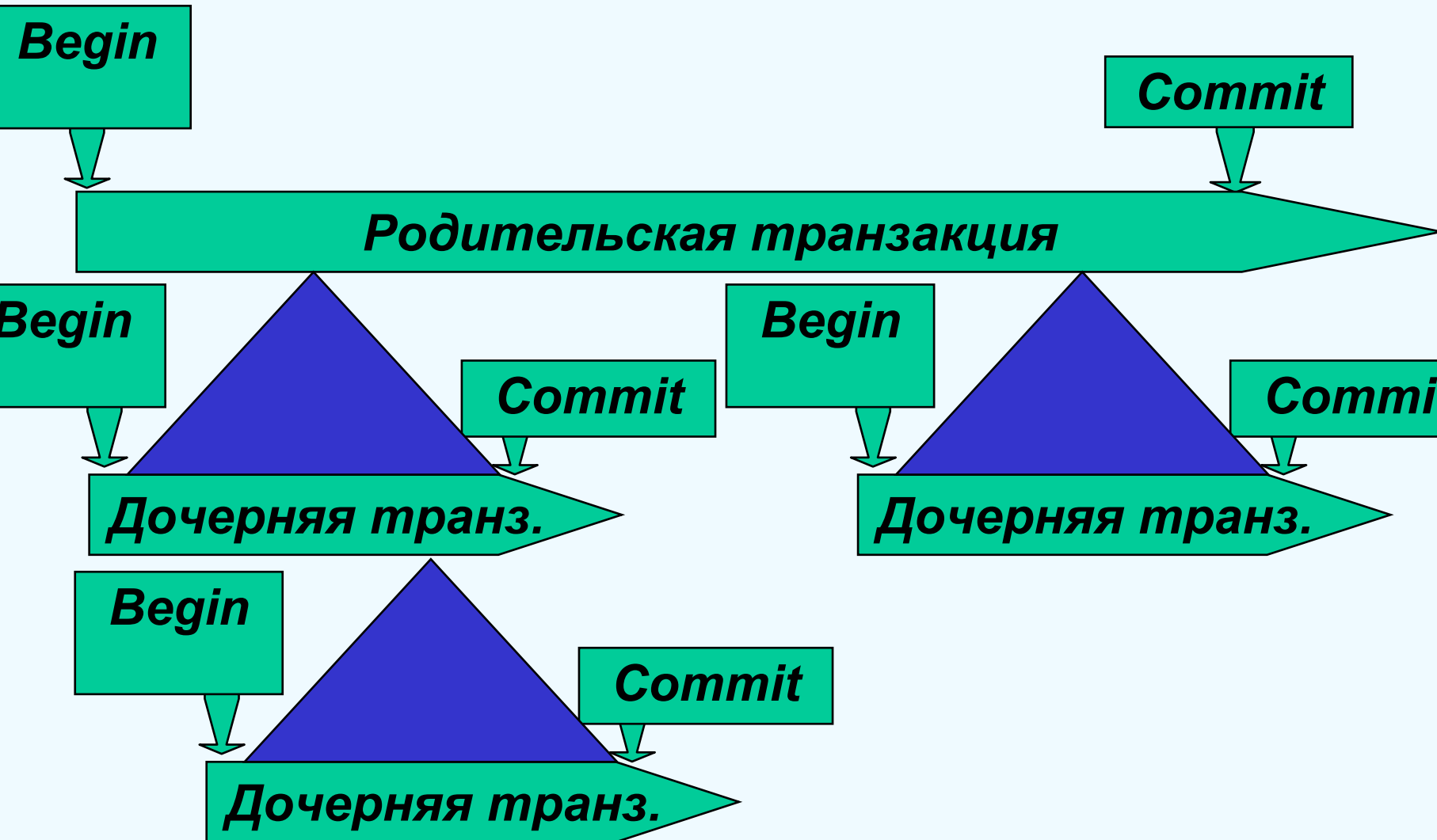
ACID

- **Атомарность (atomicity)** – транзакция либо завершена полностью, либо полностью отменена
- **Непротиворечивость (consistency)** – гарантирует, что последовательность операций оставит набор ресурсов в непротиворечивом состоянии
- **Изолированность (isolation)** – гарантирует изолированное выполнение транзакции независимо от других конкурирующих транзакций
- **Долговечность (durability)** – после завершения транзакции выполненные изменения становятся долговечными даже в случае аварийного завершения процесса

Плоские транзакции



Вложенные транзакции



Распределенные и централизованные транзакции

- При распределенных транзакциях
 - Данные содержатся в различных БД или
 - Распределенные объекты не используют БД
 - Обработка транзакций не прозрачна для разработчиков

Управление транзакциями: клиент и сервер

- В случае управления транзакциями сервером он может взять на себя заботу о начале и завершении транзакции
- В случае управления со стороны клиента серверные объекты должны реализовывать функции транзакций (начала, отмены и фиксации), а также функции совместного доступа (блокировка ресурса и т.п.)

План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Сервисы распределенных объектных транзакций

Управление совместным доступом

- Компоненты распределенных систем используют разделенные ресурсы конкурентно:
 - Компоненты технического обеспечения
 - Ресурсы ОС
 - БД
 - Объекты
- Доступ к некоторым ресурсам д.б. организован так, чтобы исключить одновременный доступ
 - Записи БД
 - Объекты CORBA

Мотивация

- Конкурентный доступ и обновления могут вести к следующим последствиям:
 - Потерянные обновления (lost updates)
 - Противоречивый анализ (inconsistent analysis)
- Пример потерянных обновлений:
 - Получение денег из банкомата и
 - Кредитование счета
- Пример противоречивого анализа:
 - Перевод средств с одного счета на другой
 - Подсчет баланса счетов

Пример

DirectBanking

+ funds_transfer(from : Account, to : Account, amount : float)

Account

- balance : float = 0

+ credit(amount : float)
+ debit(amount : float)
+ get_balance() : float

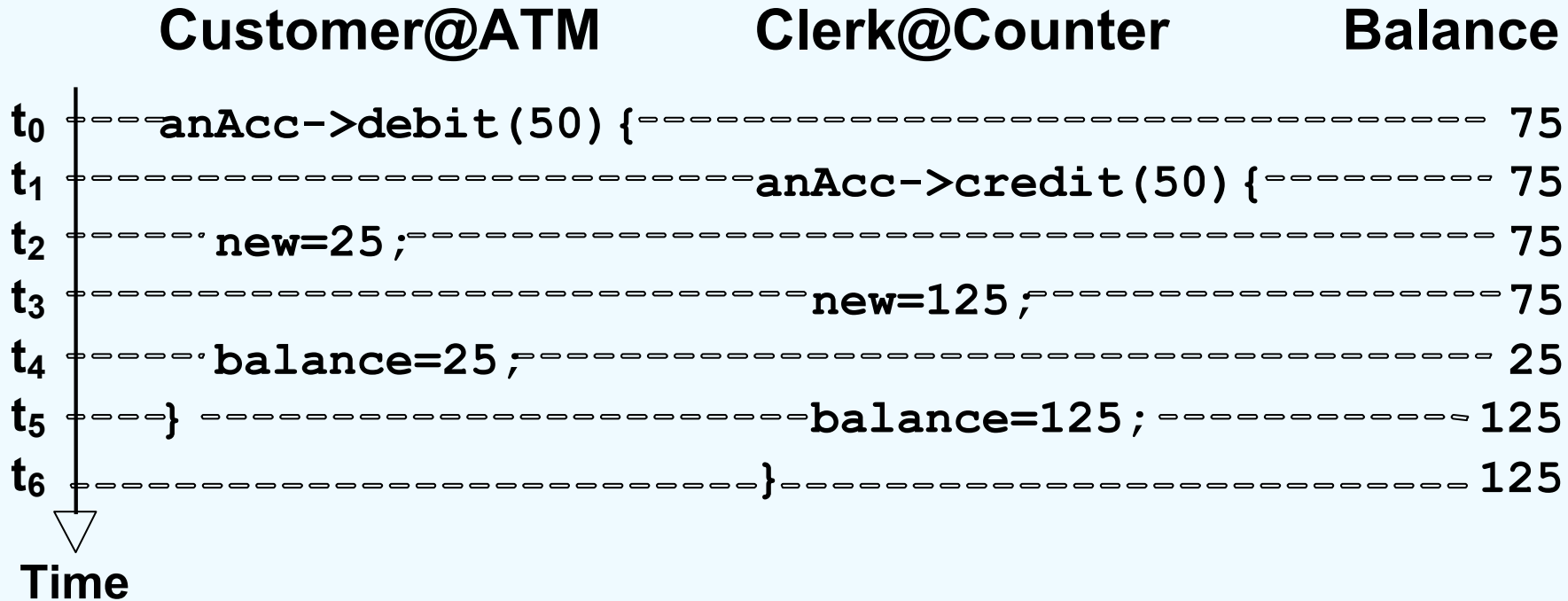
Reporting

+ sum_of_accounts(set : Account[]) : float

Пример

```
class Account {  
    protected:  
        float balance;  
    public:  
        float get_balance() {return balance;};  
        void debit(float amount) {  
            float new=balance-amount;  
            balance=new;  
        };  
        void credit(float amount) {  
            float new=balance+amount;  
            balance=new;  
        };  
};
```


Потерянные обновления



Противоречивый анализ

Funds Transfer

Inland Revenue Report

Acc1Acc2 Su

0	---Acc1->debit(7500) {	sum=0;	-----7500	----	0----
1	new=0;		-----7500	----	0----
2	balance=0;	sum+=Acc2->get_balance();	-----0	----	0----
3	}		-----0	----	0----
4	Acc2->credit(7500) {		-----0	----	0----
5	new=7500;		-----0	----	0----
6	balance=7500;	sum+=Acc1->get_balance();	-----0	7500	----
7	}		-----0	7500	----

time

Обеспечение параллельной работы

- Пессимистический подход
 - Предпосылка: параллельный доступ скорее всего произойдет
 - Блокирование ресурсов для исключения возможности параллельного доступа с помощью 2-х фазной блокировки
- Оптимистический подход
 - Предпосылка: параллельный доступ скорее всего не произойдет
 - Ведение идентификаторов строк и их контроль в случае обновления:
ROWID в SQLBase

2-х фазная блокировка

- Большинство популярных СУБД реализуют именно этот метод блокировки
- Конкурентные процессы запрашивают блокировки у менеджера блокировок
- Менеджер блокировок предоставляет блокировки, если они не конфликтуют с уже предоставленными блокировками
- Альтернатива: оптимистическое управление одновременным доступом (предполагается, что аномалий не произойдет)

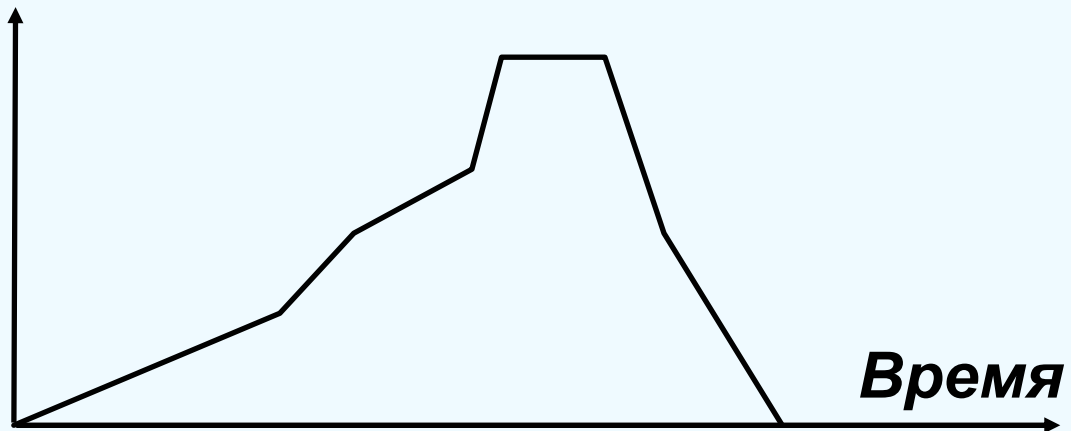
Блокировки

- Блокировка является признаком, свидетельствующим о том, что ресурс используется процессом в определенном режиме
- Минимальные режимы блокировки: чтение и запись
- При 2-х фазном блокировании процессы сначала получают блокировки, а после – освобождают

Блокировки

- Процессы получают блокировки до доступа к ресурсам и освобождают после
- Типичный профиль блокирования представлен на рисунке ниже:

**Кол-во
блокировок**



Гранулярность блокировок

- При высокой степени конкурентного доступа используется мелкогранулярные блокировки
- Мелкогранулярные блокировки приводят к необходимости использования большого количества блокировок
- Большое число блокировок ведет к высоким издержкам
- Иерархические блокировки могут служить компромиссом

Иерархические блокировки

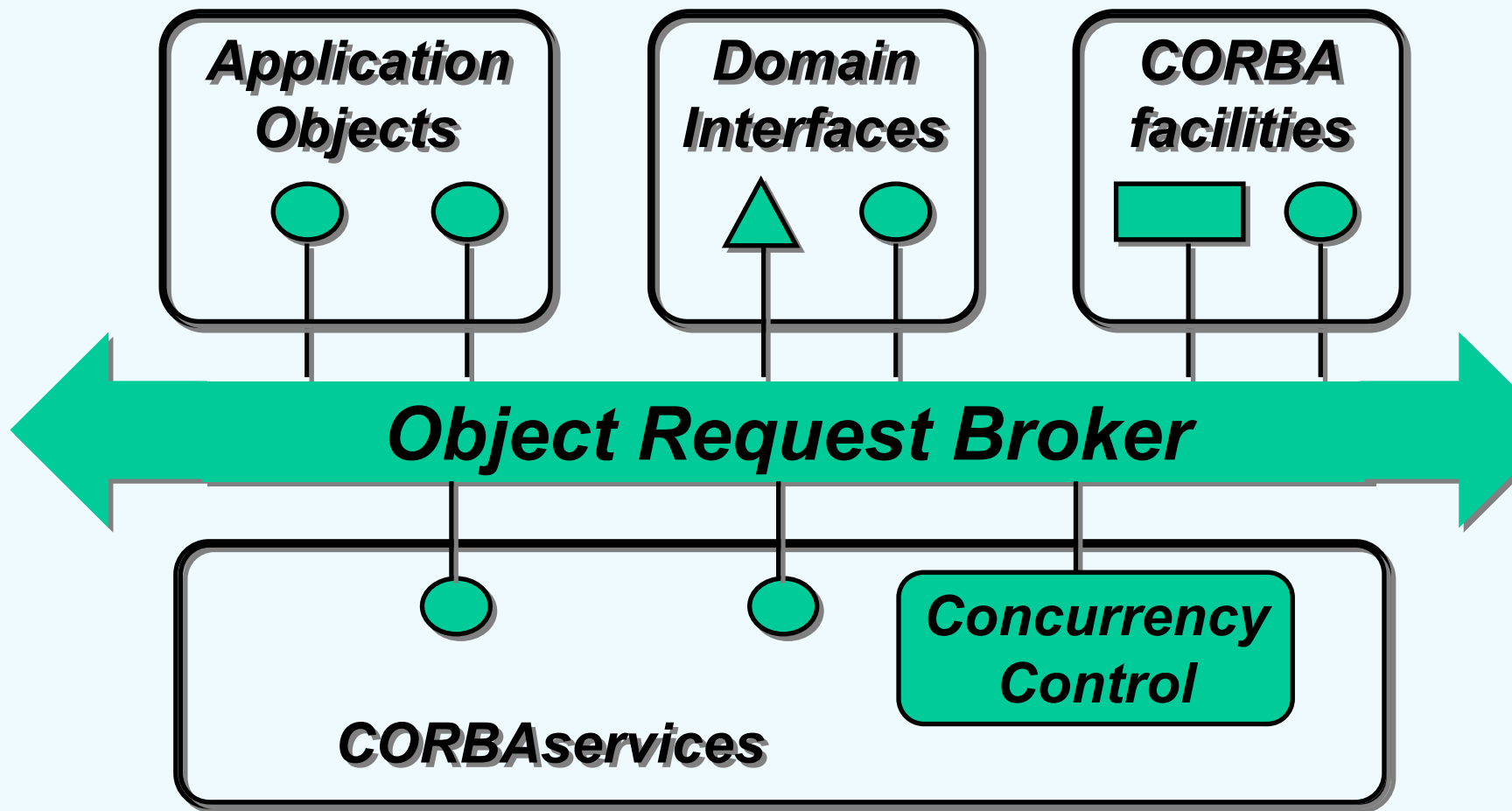
- Применяются для композитных ресурсов, например:
 - Файл
 - Набор или последовательность
- Намерение чтения или записи присваивается контейнерам
- Матрица совместимости

	IR	R	IW	W
IR	+	+	+	-
R	+	+	-	-
IW	+	-	+	-
W	-	-	-	-

План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Долговременное хранение объектов

Сервис контроля совместного доступа CORBA



Сервис контроля совместного доступа CORBA

- Реализует пессимистическую стратегию блокировок
- Спецификации сервиса контроля совместного доступа приняты в 1994
- Целью данного сервиса является обеспечение механизма обеспечивающего контроль за совместным доступом к объектам так, что они остаются в непротиворечивом состоянии
- Сервис предоставляет интерфейс для предоставления 2-х фазных блокировок

Матрица совместимости блокировок

- Поддержка иерархических блокировок
- Матрица совместимости блокировок
 - IR – намерение чтения
 - IW – намерение записи
 - U – усиление (блокировка чтения, конфликтующая сама с собой, позволяет снизить вероятность взаимной блокировки)
 - W – запись
 - R – чтение

	IR	R	U	IW	W
IR	+	+	+	+	-
R	+	+	+	-	-
U	+	+	-	-	-
IW	+	-	-	+	-
W	-	-	-	-	-

Наборы блокировок

- Набор блокировок ассоциируется с ресурсом (обычно в реализации с данным ресурсом)
- Каждый разделенный ресурс имеет набор блокировок
- Операции данного ресурса получают блокировки до того как они получают доступ или изменяют ресурс
- Для вложенных транзакций правило ослабляется: вложенные транзакции могут присваивать себе родительские транзакции

IDL интерфейс

```
interface LocksetFactory {  
    LockSet create();  
};  
  
interface Lockset {  
    void lock(in lock_mode mode);  
    boolean try_lock(in lock_mode mode);  
    void unlock(in lock_mode mode);  
    void change_mode(in lock_mode held,  
                     in lock_mode new);  
};
```

План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Сервисы распределенных объектных транзакций

2-х фазная фиксация транзакций

- Роли:
 - Транзакционный клиент
 - Транзакционный сервер
 - Координатор

Координатор

- Управляет выполнением транзакций
- Обрабатывает вызовы начала, отмены и фиксации транзакций
- Распределяет системные уникальные транзакционные идентификаторы
- Различные транзакции могут иметь различные координаторы транзакций

Транзакционный сервер

- Любой компонент с ресурсом изменяемым в рамках транзакционного управления
- Транзакционные сервер должен знать своего координатора
- Должен зарегистрировать свое участие в транзакции у координатора
- Должен реализовывать 2-х фазный протокол фиксации

Транзакционный клиент

- Взаимодействует с транзакциями через координатор
- Вызывает сервисы координатора для начала, завершения и отмены транзакций
- Реализация транзакций прозрачна для клиентов

2-х фазная фиксация

- Множество распределенных автономных серверов
- Для общей фиксации все серверы должны быть готовы к фиксации
- Если один из серверов не может зафиксировать изменения, то все серверы должны отменить фиксацию
- Однофазный протокол не достаточен в этом случае
- Необходимо использовать 2 фазы:
 - Голосование
 - Завершение

2-х фазная фиксация: фаза 1

- Фаза голосования
- Координатор спрашивает могут ли все серверы фиксировать транзакцию
- Если да, то отвечает «ДА»
- Если нет, то отменяет транзакцию и отвечает «НЕТ»
- Таким образом сервер может самостоятельно отменить, но не может самостоятельно зафиксировать транзакцию

2-х фазная фиксация: фаза 2

- Фаза завершения
- Координатор собирает все голоса и если все положительны, то командует «зафиксировать», если нет, то «отменить»
- Серверы подтверждают команду «зафиксировать» по завершению таковой

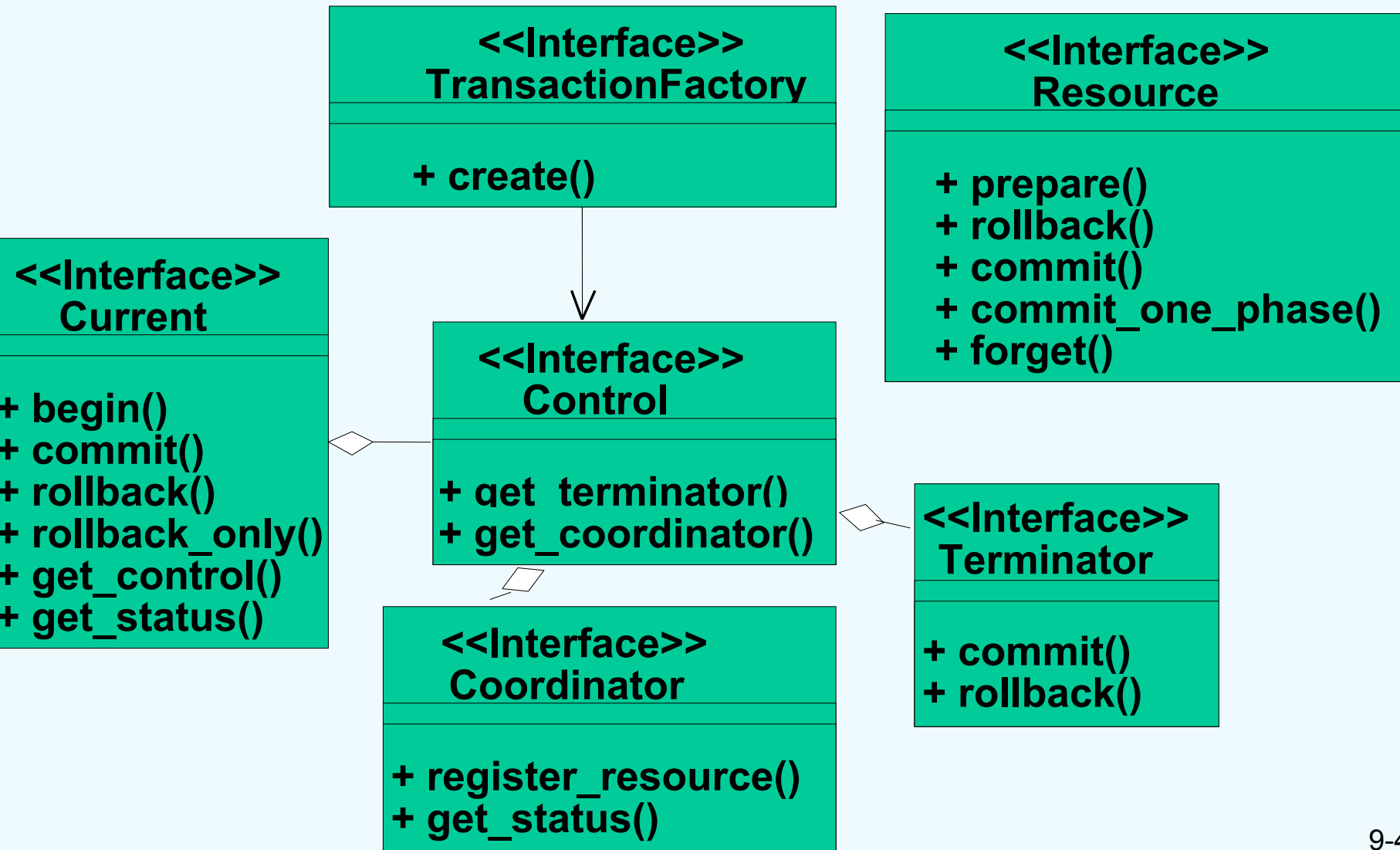
План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Сервисы распределенных объектных транзакций

Сервис объектных транзакций CORBA

- OTS – Object Transaction Services (Сервис объектных транзакций)
- Определен в 1994 году в качестве части набора сервисов CORBA
- Данный сервис позволяет:
 - Сформировать серию вызовов и обращений к объектам в единую транзакцию, обеспечивая атомарность и долговечность
 - Организовать вложенные транзакции
- Сервис объектных транзакций опирается на сервис контроля одновременного доступа для обеспечения свойства изоляции

Интерфейсы IDL



Интерфейс Current

```
interface Current {  
    void begin() raises (...);  
    void commit (in boolean report_heuristics)  
        raises (NoTransaction, HeuristicMixed,  
                HeuristicHazard);  
    void rollback() raises (NoTransaction);  
    Status get_status();  
    string get_transaction_name();  
    Coordinator get_control();  
    Coordinator suspend();  
    void resume(in Coordinator which)  
        raises (InvalidControl);  
};
```

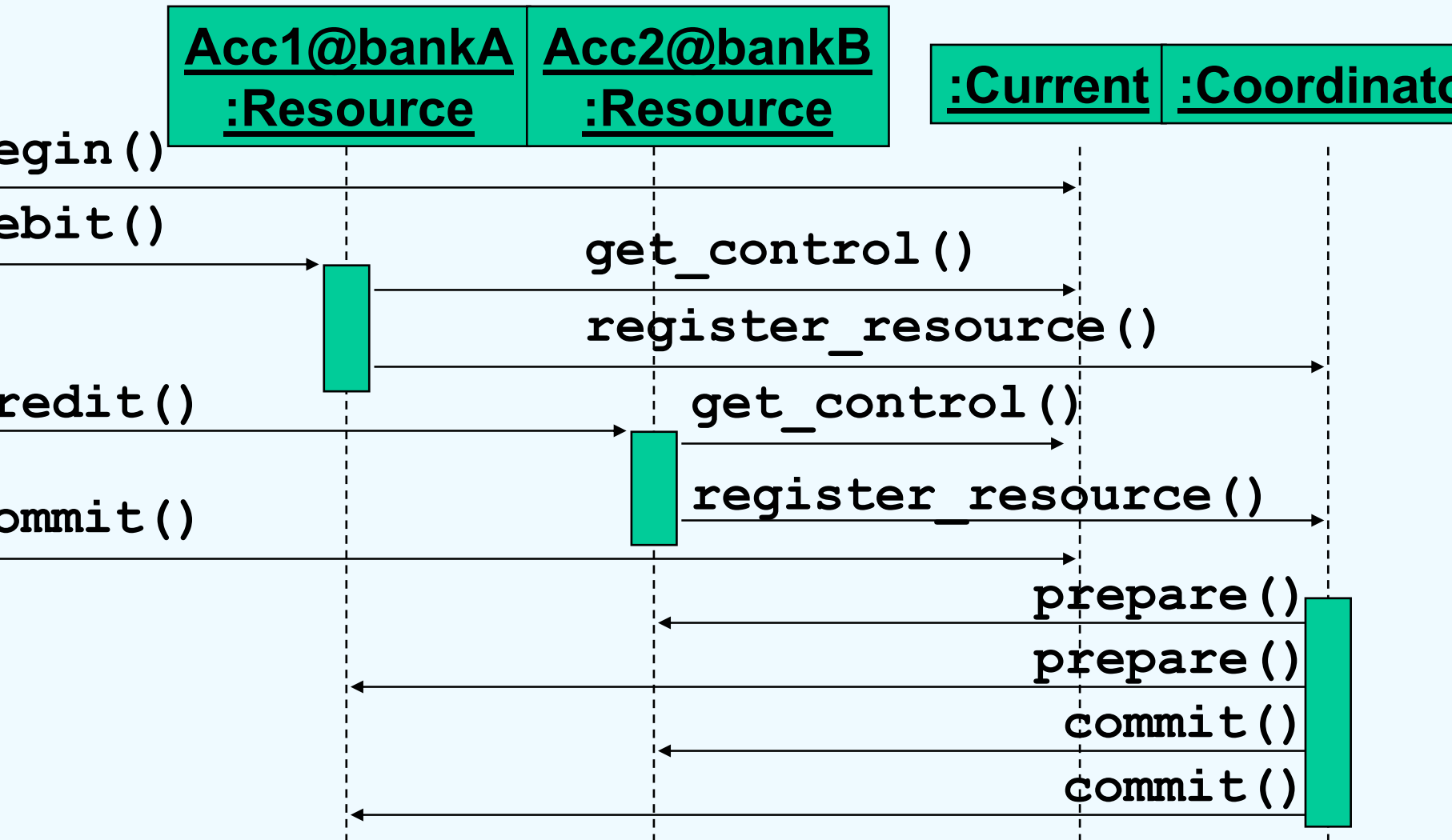
Интерфейс Coordinator

```
interface Coordinator {  
    Status get_status();  
    Status get_parent_status();  
    Status get_top_level_status();  
    boolean is_same_transaction(in Coordinator tr);  
    boolean is_related_transaction(in Coordinator tr);  
    RecoveryCoordinator register_resource(  
        in Resource r) raises(Inactive);  
    void register_subtran_aware(  
        in SubtransactionAwareResource r)  
        raises(Inactive, NotSubtransaction);  
    ...  
}
```

Интерфейс Resource

```
enum Vote {  
    VoteCommit,  
    VoteRollback,  
    VoteReadOnly  
};  
  
interface Resource {  
    Vote prepare();  
    void rollback() raises(...);  
    void commit() raises(...);  
    void commit_one_phase raises(...);  
    void forget();  
};
```

Пример перевода средств

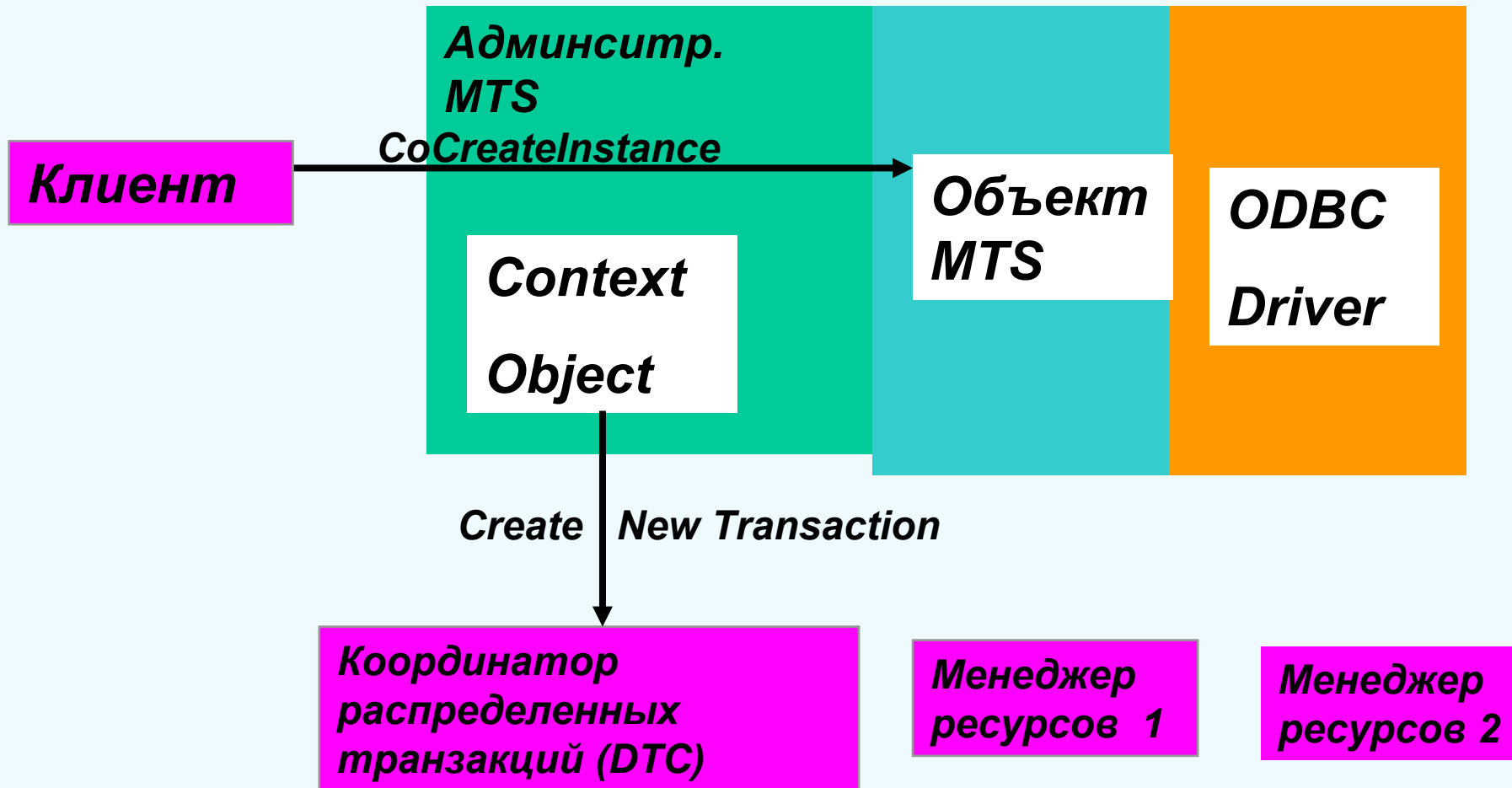


План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Сервисы распределенных объектных транзакций

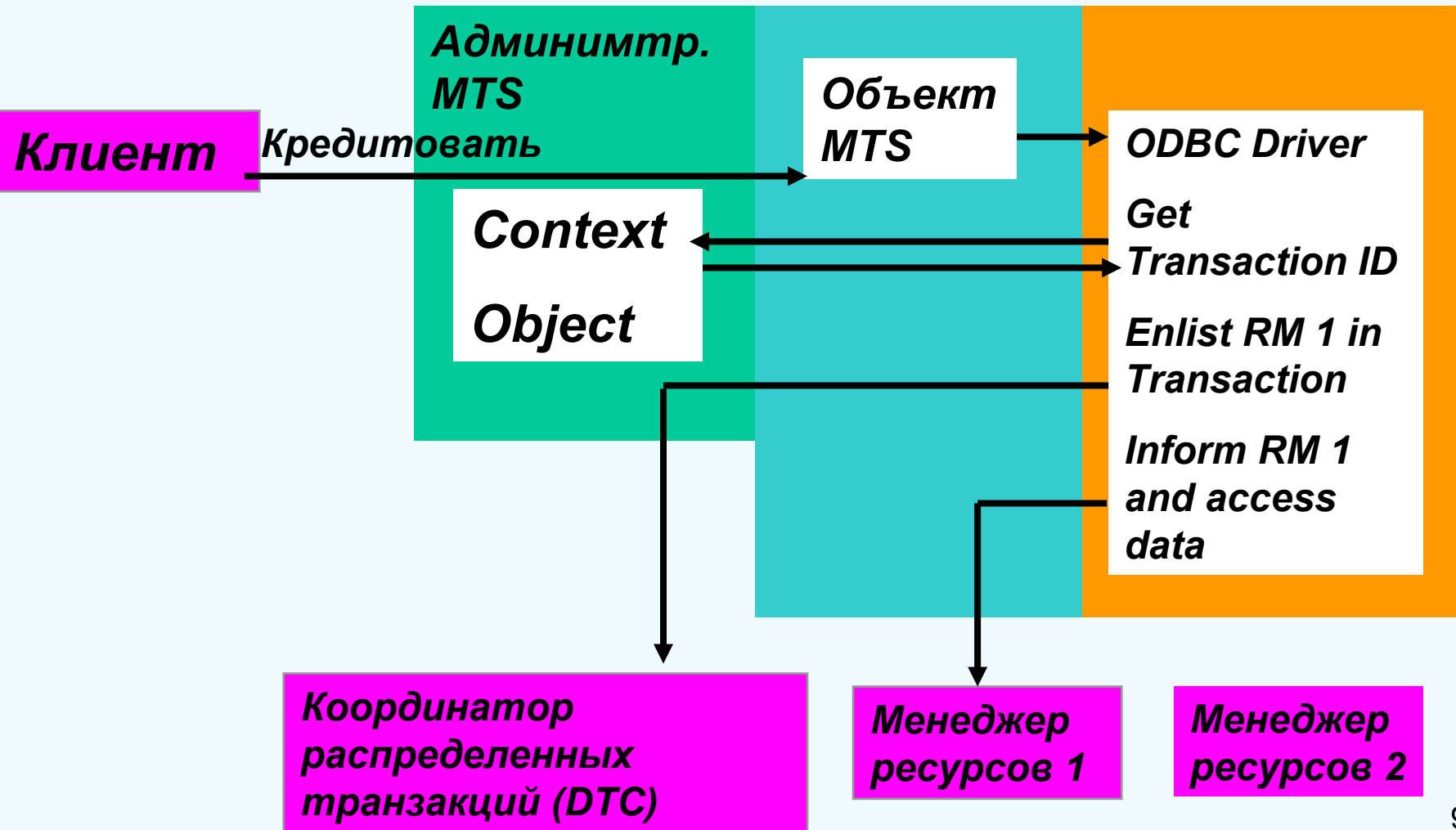
Транзакции в MTS

Этап 1



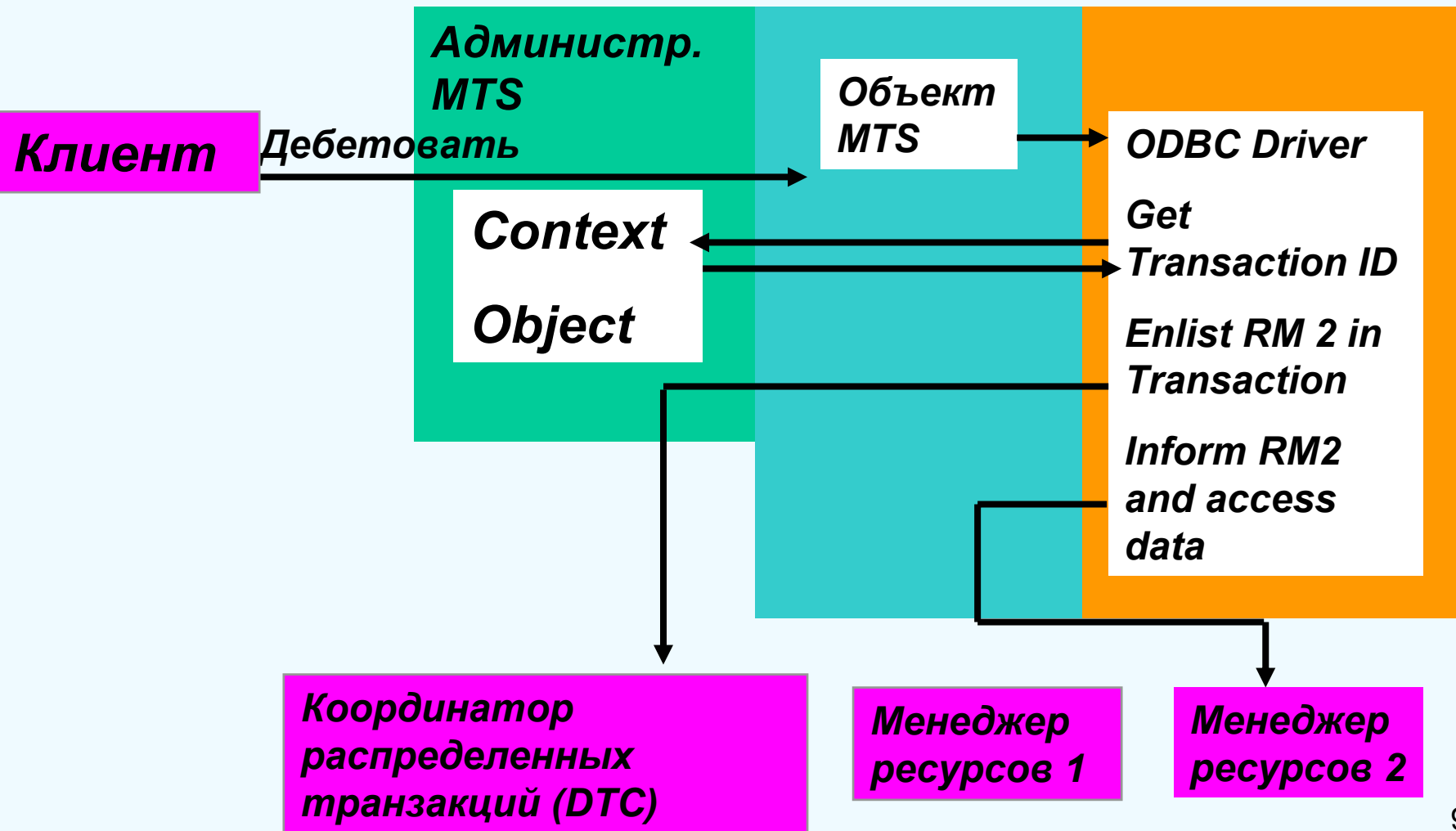
Транзакции в MTS

Этап 2



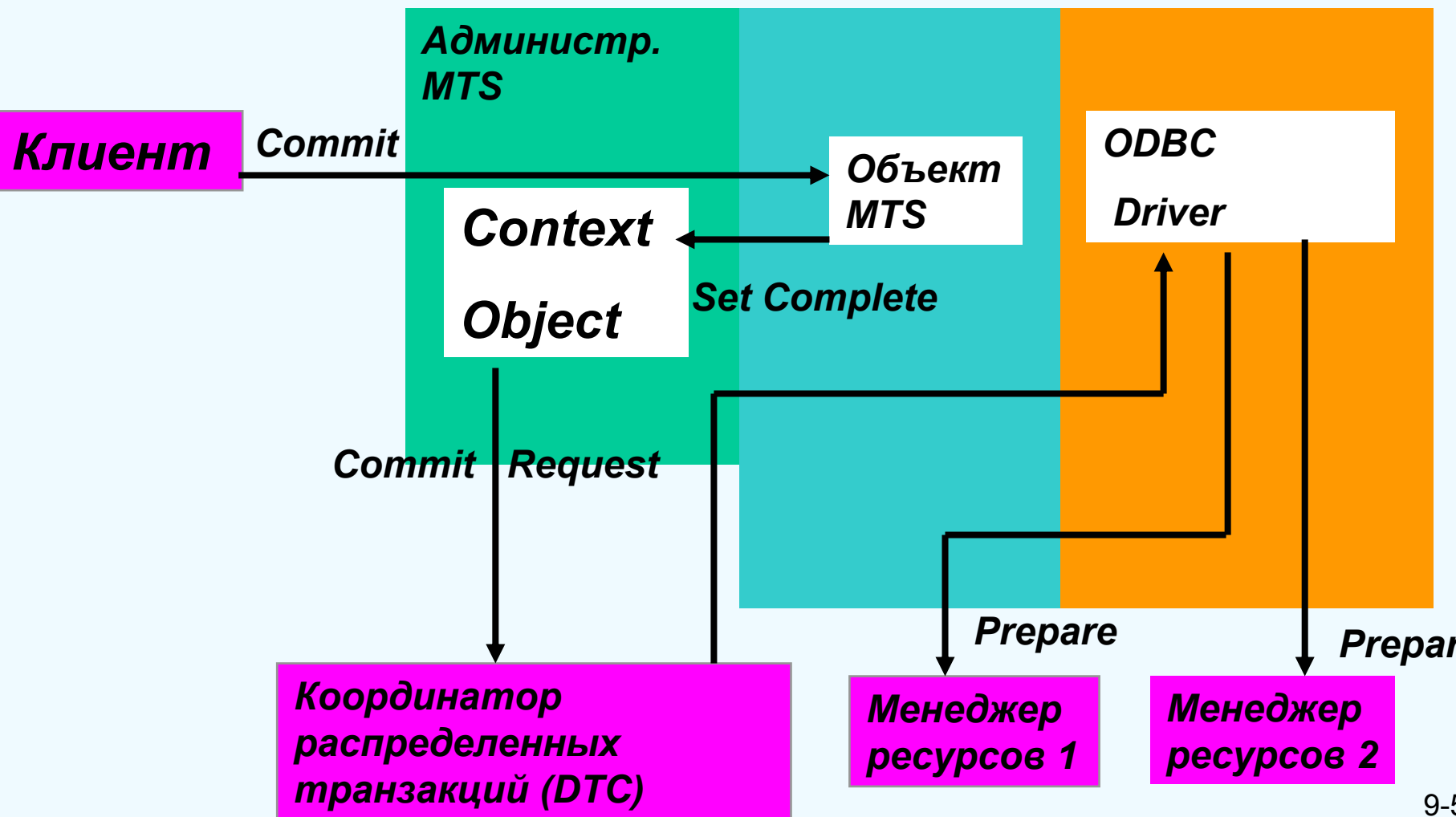
Транзакции в MTS

Этап 3



Транзакции в MTS

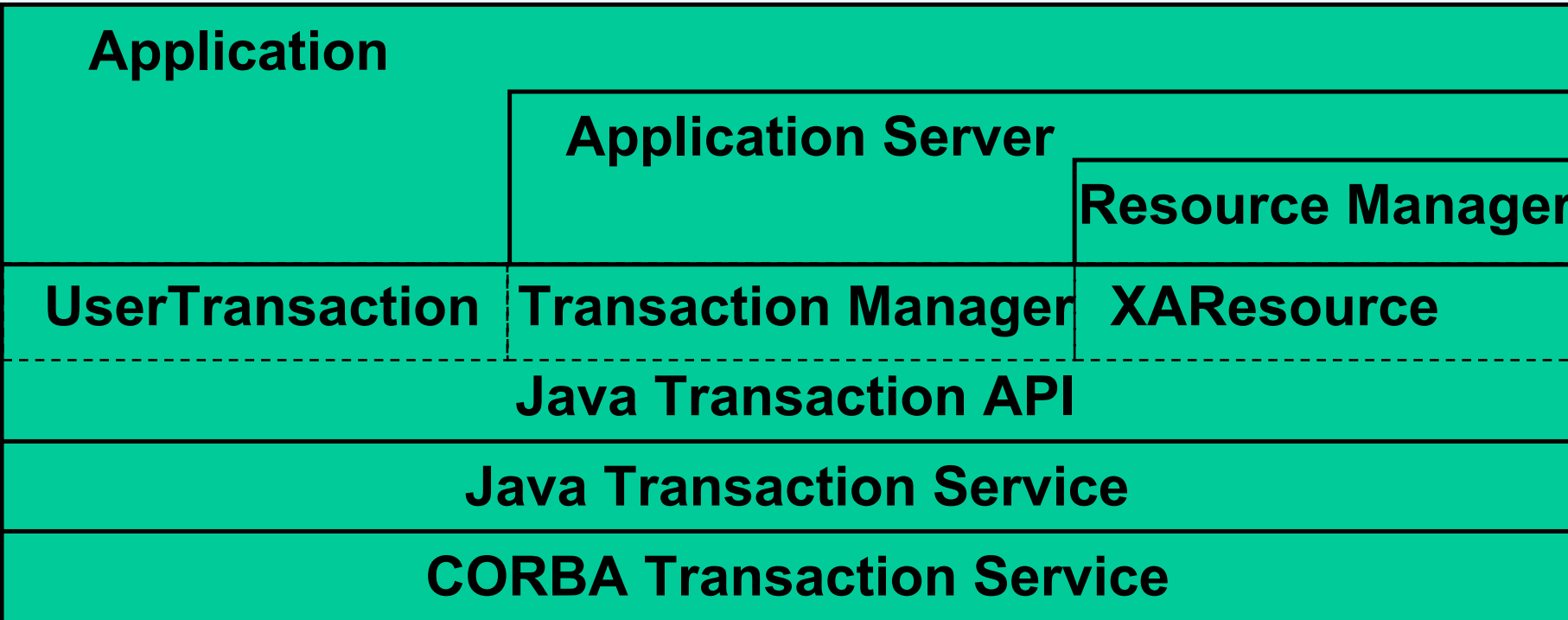
Этап 4



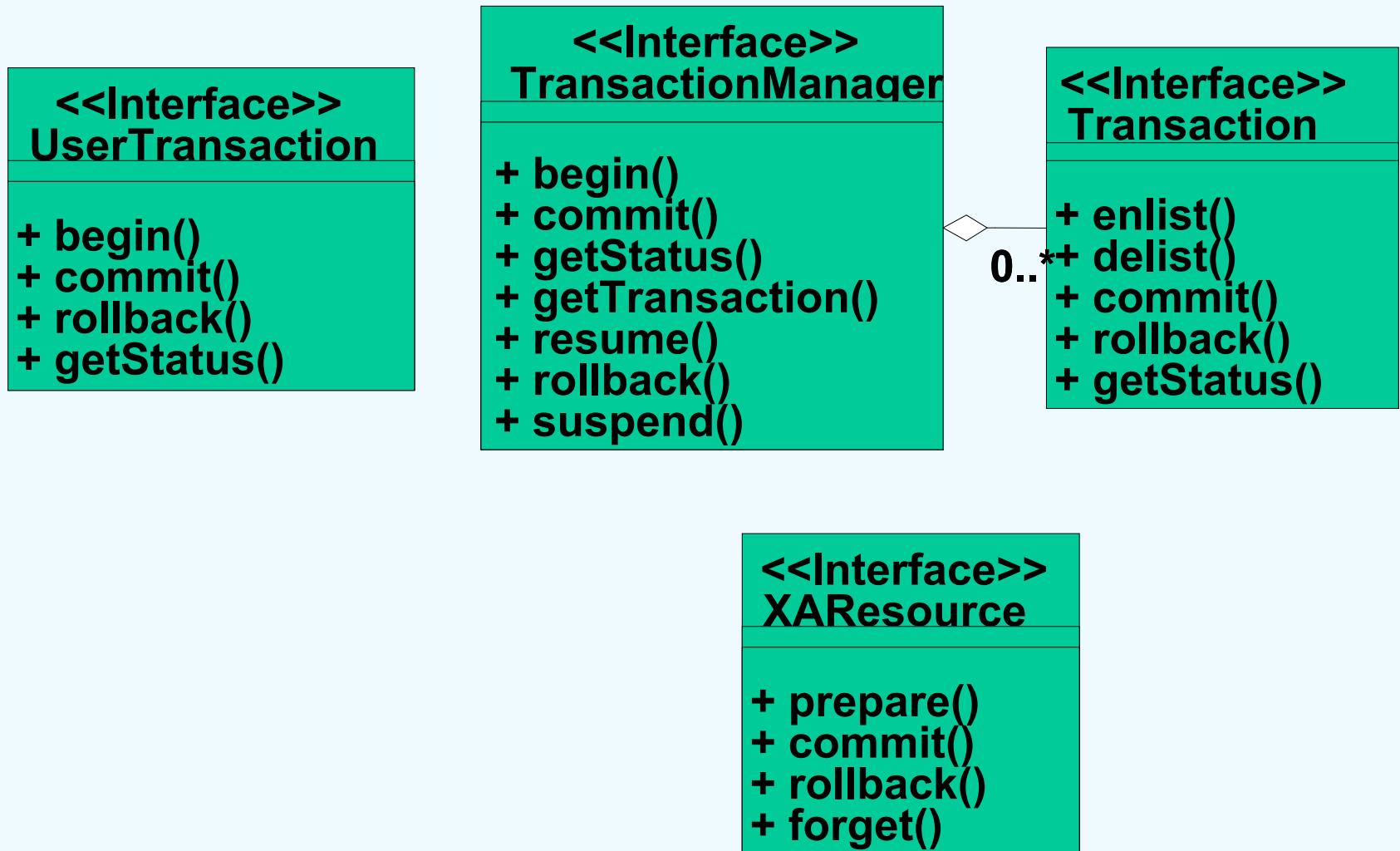
План лекции

- Принципы транзакций
- Управление одновременным выполнением
- Протокол двухфазной фиксации
- Сервисы распределенных объектных транзакций

Управление транзакциями в Java



Транзакционный интерфейс Java



Ключевые моменты

- Распределенная объектная транзакция – атомарная, сохраняющая непротиворечивость, изолированная и долговечно хранимая последовательность объектных запросов
- Объекты, принимающие участие в распределенных объектных транзакциях, выполняют роль транзакционных клиентов, серверов и координаторов
- Изоляция достигается с помощью двухфазной блокировки и может быть либо делегирована СУБД, либо явно обеспечена на уровне объектов

Ключевые моменты

- Атомарность достигается с помощью двухфазной фиксации транзакции, состоящей из фаз голосования и фиксации

Литература / Internet ИСТОЧНИКИ

- В. Эммерих *Конструирование распределенных объектов*. - М.:Мир. - 2002.
- Vinoksi S. ***CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments***, IEEE'96.
- Schmidt D.C. And Vinoski S. ***Object Adapters: Concepts and Technology***, SIGS C++ Report, Vol. 9, No. 11, Nov-Dec 1997.
- Ю.А. Григорьев, А.Д. Плутенко. ***Жизненный цикл проектирования распределенных баз данных***. - Благовещенск. - 1999.
- www.omg.org